NBSIR 86-3473

The Internal Revenue Service Post-Of-Duty Location Modeling System - Programmer's Manual for Fortran Driver

Paul D. Domich, Richard H. F. Jackson, Marjorie A. McClain

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Applied Mathematics
Gaithersburg, MD 20899

July 1986

Issued February 1987

NBSIR 86-3473

THE INTERNAL REVENUE SERVICE POST-OF-DUTY LOCATION MODELING SYSTEM - PROGRAMMER'S MANUAL FOR FORTRAN DRIVER

Paul D. Domich, Richard H. F. Jackson, Marjorie A. McClain

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Applied Mathematics
Gaithersburg, MD 20899

July 1986

Issued February 1987

Sponsored by:
The Research Division
Internal Revenue Service
Washington, DC 20224

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

ABSTRACT

This report is a programmer's manual for a microcomputer package which was designed by the National Bureau of Standards to assist the Internal Revenue Service in choosing locations for its posts-of-duty which will minimize costs to the IRS and to the taxpayer. The package was written in two sections of code, one in FORTRAN and the other in PASCAL. This manual describes the FORTRAN driver which handles graphics displays and controls input and output for the solution procedure.

Keywords: facility location, interactive graphics, personal computer, microcomputer, Graphical Kernel System (GKS)

TABLE OF CONTENTS

Section I: Introduction

The Internal Revenue Service Post-of-Duty Location System is a microcomputer package designed to assist IRS district planners in choosing locations for posts-of-duty (POD's) which will minimize costs to the IRS and to the taxpayer. The package uses color graphics in performing tasks such as displaying maps of workload, setting user options to initialize location problems, and displaying solutions to location problems.

This manual is one of a series of reports documenting the POD location system. The reports in the series are as follows.

1) The Internal Revenue Service Post-of-Duty Location Modeling System: Final Report

   This report describes the post-of-duty location problem and its mathematical model. It discusses the types of data which are considered in calculating costs, describes the methods used to solve the location problem, and gives a brief introduction to the computer implementation of the model. (NBS Contact: Richard H. F. Jackson)

2) The Internal Revenue Service Post-of-Duty Location Modeling System: User's Manual

   This report is a user's guide for the post-of-duty location computer system. It gives hardware and software requirements, instructions for installing the system, descriptions of data files, and detailed instructions for operating the system. (NBS Contact: Marjorie A. McClain)

3) The Internal Revenue Service Post-of-Duty Location Modeling System: Programmer's Manual for FORTRAN Driver

   The post-of-duty location program is written in two sections of code, one in FORTRAN and the other in PASCAL. This report describes the FORTRAN driver which handles graphics displays and controls input and output for the solution procedure. The report includes an alphabetical list of the FORTRAN routines, describing the purpose, the calling sequence and the variables of each routine. (NBS Contact: Marjorie A. McClain)

4) The Internal Revenue Service Post-of-Duty Location Modeling System: Programmer's Manual for PASCAL Solver

   This report describes the second part of the post-of-duty location program, the PASCAL solver. It discusses the algorithms and data structures used to solve a location problem. (NBS Contact: Paul D. Domich)

It is assumed that the reader of this programmer's manual is thoroughly familiar with IBM-PC DOS, with the FORTRAN language, and with the POD location system User's Manual.

Note: Reference to a tradename or product in this report does not imply endorsement by the National Bureau of Standards.

Section II: Software Requirements

The following software is required to be able to use and make changes to the POD location system programs.

1) IBM-PC DOS (Version 2.1 or later)
2) IBM Professional FORTRAN (Version 1.00)
3) Turbo PASCAL (Turbo-87 Version 3.0)
4) IBM Graphical Kernel System (Version 1.00)
   (GKS includes the Virtual Device Interface and device drivers. See the User's Manual for information on how to set up the device drivers in the AUTOEXEC.BAT and CONFIG.SYS files.)
5) Source code for the POD location system, contained in the following files.
   Data File Initialization Programs --
      TEMPZIP.BAS
      UNFORM.FOR
      SAVCEN.FOR
      TEMPSTE.BAS
      BINSTE.FOR
   System Driver Programs --
      LOCATE.BAT
      DRIVER.FOR
      IRS.FOR
      GKSUTIL.FOR
   System Solver Programs --
      SOLVER.PAS
      INIT.PAS
      DSTRUCT.PAS
      GREEDY.PAS
      INTCHG.PAS
      PODCLR.PAS
      FIVCLR.PAS
6) Executable code for the POD location system, contained in the following files.
   Data File Initialization Programs --
      TEMPZIP.BAS
      UNFORM.EXE
      SAVCEN.EXE
      TEMPSTE.BAS
      BINSTE.EXE
   System Driver Programs --
      LOCATE.BAT
      DRIVER.EXE
   System Solver Program --
      SOLVER.COM

Only a subset of this software is required for a user who just wants to run the system and not make any changes to it. See the User's Manual for user software requirements and also for general hardware requirements.

Section III: Description of Data File Initialization Programs

The POD location system requires the use of zip code boundary coordinates obtained from a commercial vendor. This data must be processed when the system is first installed to convert it to a useable format. The following program files are used in the initialization process.

1) TEMPZIP.BAS

This BASIC program reads a zip code boundary file from floppy disks and writes it onto a fixed disk. In the process, labels are removed and end-of-record marks are added to make the file readable by a FORTRAN program.

2) UNFORM.FOR

This FORTRAN program reads the file created by TEMPZIP.BAS and converts it to an unformatted direct access file. This is done to allow faster data accessing.

3) SAVGEN.FOR

This FORTRAN program calculates the centroid of each zip code by taking a weighted average of its boundary points. The centroids are stored in an unformatted direct access file which also contains five-digit zip code names and pointers to each zip code in the boundary file.

4) TEMPSTE.BAS

This BASIC program reads a state boundary coordinate file from a floppy disk and writes it onto a fixed disk. End-of-record marks are added.

5) BINSTE.FOR

This FORTRAN program reads the file created by TEMPSTE.BAS and converts it to an unformatted direct access file. Also, the state boundary coordinates are transformed to be in the same units as the zip code boundary coordinates.

Note: These coordinate files will eventually be replaced by files from another vendor, so the initialization programs will be completely rewritten.

Section IV: Overview of System Programs

The POD location system is written in two separate sections of code. One section is written in FORTRAN and contains the code for drawing maps and initializing location problems. It allows the user to display workload, set up a location problem, and display the solution. The second section is written in PASCAL and contains the code for solving a location problem. Data is passed between the two sections of code by the use of files.

A batch file called LOCATE.BAT controls the flow of execution between the two sections of code. LOCATE.BAT first checks to see if files called ERRORS.GKS and EXITFILE.BAT exist from a previous run. If so, it erases them. Then LOCATE.BAT enters a loop which passes control between the FORTRAN section (stored in DRIVER.EXE) and the PASCAL section (stored in SOLVER.COM). DRIVER is always entered first. Then, depending on actions taken by the user, either SOLVER is entered or an exit is taken from the loop. In the first case, after SOLVER has completed, control returns to DRIVER and the loop repeats. In the second case, a file called EXITFILE.BAT is created by DRIVER. It contains instructions for deleting work files created during the run. This new batch file is executed, and then LOCATE.BAT terminates.

The FORTRAN source code is contained in three files: DRIVER.FOR, IRS.FOR and GKSUTIL.FOR. DRIVER.FOR contains the main program DRIVER along with most of the high-level subroutines. IRS.FOR contains mostly intermediate-level subroutines for drawing menus and maps. GKSUTIL.FOR contains low-level subroutines for working with GKS. A complete description of each subroutine is given in Section V.

If a change is made in a subroutine, it must be recompiled by typing "PROFORT filename", where "filename" is the name of the source code file containing the subroutine. Then the program must be relinked by typing "LINK DRIVER IRS GKSUTIL \GKS\PFGKS,,,\PROFORT\PROFORT \GKS\GKS \GKS\PFGKS /S:5000 /X:350". (This assumes that DRIVER.OBJ, IRS.OBJ and GKSUTIL.OBJ are stored in the current directory, Professional FORTRAN files are in a directory called \PROFORT and GKS files are in a directory call \GKS. These path names may be changed.) The result is a new executable file called DRIVER.EXE.

The PASCAL solver routines are described in "The Internal Revenue Service Post-of-Duty Location Modeling System: Programmer's Manual for PASCAL Solver".

Section V: Description of FORTRAN Routines

This section contains brief descriptions of each of the FORTRAN routines. Information is provided on the purpose of the routine, the name of the file containing its source code, names of input and output variables, and names of programs called by the routine and programs which call it. The routines are listed in alphabetical order.

SUBROUTINE BEEP:

This subroutine beeps the speaker.

Source Code Location --
GKSUTIL.FOR

Programs Called --
None

Calling Programs --
STEMAP
ZIPMAP

SUBROUTINE BORDER(XMIN,XMAX,YMIN,YMAX):

This subroutine draws a border around the current screen window. The provided window limits are slightly reduced before the border is drawn; otherwise some sides of the border may not appear because of roundoff error in the conversion from world coordinates to screen coordinates. (See the GKS manual for definitions of graphics terms.)

Source Code Location --
GKSUTIL.FOR

Input Variables --
XMIN: REAL*4
Smallest x value of world coordinates
XMAX: REAL*4
Largest x value of world coordinates
YMIN: REAL*4
Smallest y value of world coordinates
YMAX: REAL*4
Largest y value of world coordinates

Programs Called --
BOX
GSPLCI

Calling Programs --
ERSMNU

MAPKEY
MENU20
MENU21
MENU22
MENU23
SPLTWN
TOPMNU
ZIPMAP

SUBROUTINE BOX(XMIN,XMAX,YMIN,YMAX):

This subroutine draws a box. (See the GKS manual for definitions of graphics terms.)

Source Code Location --
GKSUTIL.FOR

Input Variables --
XMIN: REAL*4
X coordinate of lower left corner (in world coordinates)
XMAX: REAL*4
X coordinate of upper right corner (in world coordinates)
YMIN: REAL*4
Y coordinate of lower left corner (in world coordinates)
YMAX: REAL*4
Y coordinate of upper right corner (in world coordinates)

Program Called --
GPL

Calling Programs --
BORDER
MENU20
MENU21
MENU23

SUBROUTINE CENSRT(IERR,STATE,MAXM,M,XCENT,YCENT,INDEX,PNT,ZIP5):

This subroutine reads the centroid file STATEXX.CEN and calls a heapsort program to sort the centroids according to increasing x and y coordinates. (This is done to speed up the process of searching for zip codes on a map. It also speeds up the drawing of maps.) Arrays are set up to store the following information from the file: centroids, pointers to the zip code boundary file, zip code index numbers, and five-digit zip code names. This subroutine is executed whenever the program DRIVER is entered.

Source Code Location --
IRS.FOR

Input Variables --

```
STATE:     CHARACTER*2
           State code number
MAXM:      INTEGER*4
           Maximum number of zip codes allowed

Output Variables --
IERR:      INTEGER*4
           Error flag --
           IERR=0 for normal return
           IERR=1 if an error was encountered
M:         INTEGER*4
           Number of zip codes
XCENT(M): REAL*4
           Array containing x coordinates of centroids,
           sorted according to increasing x and y
YCENT(M): REAL*4
           Array containing y coordinates of centroids,
           sorted according to increasing x and y
INDEX(M): INTEGER*2
           Zip code index array --
           INDEX(I) is the original index of the zip code with
           centroid (XCENT(I),YCENT(I)) before sorting
PNT(M):    INTEGER*2
           Pointer array --
           PNT(I) points to the beginning of information on the
           zip code with centroid (XCENT(I),YCENT(I)) in the zip
           code boundary file
ZIP5(M):   INTEGER*4
           Array of five-digit zip codes, in original sequential order

Program Called --
SORT

Calling Program --
DRIVER
```

SUBROUTINE CHOICE(WKID,CHDNR):

This subroutine initializes choice mode (function keys). To use the function keys, call GRQCH ("REQUEST CHOICE"). (See the GKS manual for definitions of graphics terms.)

```
Source Code Location --
GKSUTIL.FOR

Input Variable --
WKID:  INTEGER*2
       Workstation identifier

Output Variable --
CHDNR: INTEGER*2
       Choice device number
```

Program Called --
GSCHM

Calling Programs --
DISPLY
DRIVER

SUBROUTINE CLSGKS(WKID):

This subroutine closes GKS and resets the screen mode for text. (This requires the presence of ANSI.SYS in the CONFIG.SYS file. See the GKS manual for definitions of graphics terms.)

Source Code Location --
GKSUTIL.FOR

Input Variable --
WKID: INTEGER*2
Workstation identifier

Programs Called --
GCLKS
GCLWK
GDAWK

Calling Programs --
DISPLY
DRIVER
MAPKEY
STEMAP
STYLE
ZIPMAP

FUNCTION COSTFN(IMFFNC,WRK,IRSWT,IRSFCT,TXPWT,TXPFCT,MILCST, TRVDIF,DIST,SQFCST,SQFT):

This function calculates the cost of assigning a zip code to a POD site. The cost is made up of travel costs and office space costs. Travel costs are based on the Individual Master File (IMF) workload generated by the zip code, its distance from the POD site, and the cost per mile. (Business Master File (BMF) data is not yet available.) However, the cost may be weighted by several user-supplied factors. Office space costs are based on the number of IMF returns generated by the zip code and the rental cost per square foot for office space at the POD site. (Warning: Office space costs are currently being computed incorrectly, since the required data is not yet available. The number of IMF returns examined is being used in place of the total number of IMF returns.) For further information on how costs are computed, see "The Internal Revenue Service Post-of-Duty Location Modeling System: Final Report".

```
Source Code Location --
DRIVER.FOR

Input Variables --
IMFFNC(4):  INTEGER*4
            Array indicating which IRS functions are to be included in
            the cost calculation for IMF data --
            IMFFNC(1): Examination
            IMFFNC(2): Collection
            IMFFNC(3): Taxpayer Service
            IMFFNC(4): Criminal Investigation
WRK(16):    INTEGER*4
            Workload array for current zip code
IRSWT:      REAL*4
            Weight (between 0 and 1) to be assigned to IRS travel costs
IRSFCT(16): REAL*4
            Array of IRS trip factors
TXPWT:      REAL*4
            Weight (between 0 and 1) to be assigned to taxpayer travel
            costs
TXPFCT(16): REAL*4
            Array of taxpayer trip factors
MILCST:     REAL*4
            Mileage cost ($/mile)
TRVDIF:     REAL*4
            Travel difficulty factor for trips between current zip code
            and current POD site
DIST:       REAL*4
            Distance between current zip code and current POD site
            (miles)
SQFCST:     REAL*4
            Office space cost for current POD site
            ($/square foot/month)
SQFT:       REAL*4
            Office space required per IMF return (square feet)

Output Variable --
COSTFN:     REAL*8
            Cost of assigning current zip code to current POD site ($)

Programs Called --
None

Calling Program --
SOLVE
```

SUBROUTINE CRSBOX(WKID,TRN,STAT,XMIN,XMAX,YMIN,YMAX):

This subroutine creates a cursor box. To use the cursor, first move the crosshair cursor to one corner of the desired box and enter this point. Then a rectangular cursor will appear which can be used to enter the

opposite corner of the box. (See the GKS manual for definitions of graphics terms.)

```
Source Code Location --
GKSUTIL.FOR

Input Variables --
WKID: INTEGER*2
      Workstation identifier
TRN:  INTEGER*2
      Transformation number

Output Variables --
STAT: INTEGER*2
      Cursor status indicator
      STAT=0: error reading cursor location
      STAT=1: no errors encountered
XMIN: REAL*4
      X coordinate of lower left corner of box
XMAX: REAL*4
      X coordinate of upper right corner of box
YMIN: REAL*4
      Y coordinate of lower left corner of box
YMAX: REAL*4
      Y coordinate of upper right corner of box

Programs Called --
GINLC
GPREC
GQNT
GRQLC

Calling Program --
ZOOMIN
```

SUBROUTINE CURSOR(WKID,TRN,IPX,IPY):

This subroutine initializes a graphics cursor. To use the cursor, call GRQLC ("REQUEST LOCATOR"). (See the GKS manual for definitions of graphics terms.)

```
Source Code Location --
GKSUTIL.FOR

Input Variables --
WKID: INTEGER*2
      Workstation identifier
TRN:  INTEGER*2
      Transformation number
IPX:  REAL*4
      Initial x position of cursor in world coordinates
IPY:  REAL*4
```

Initial y position of cursor in world coordinates

Programs Called --
GINLC
GPREC

Calling Program --
MAPKEY

SUBROUTINE DISPLY(IERR,STATE,PALETT,NCLRS,MENU,COLOR,MODIFY, M,XCENT,YCENT,INDEX,PNT,ZIPCLR):

This subroutine is the driver for displaying maps. It uses GKS to draw a state map and then allows the user to zoom in or modify the colors on the map. The following common block is required by GKS:

COMMON /GRACOM/SIZE,INTARY.

SIZE is an INTEGER*4 variable set equal to 2500, and INTARY is an INTEGER*4 array of length 2500.

Source Code Location --
IRS.FOR

Input Variables --

STATE: CHARACTER*2
State code number

PALETT: INTEGER*2
Palette identifier (1 or 2)
PALETT=1 for a variety of colors
PALETT=2 for shades of green
(PALETT is used only on an enhanced display)

NCLRS: INTEGER*2
Number of colors in menu
(only used if MODIFY=.TRUE.)

MENU: INTEGER*2
Menu type switch (for map keys) --
MENU=0 for general coloring menu
MENU=1 for menu of POD types
MENU=2 for title of solution map
MENU=3 for key to workload map

COLOR: LOGICAL
Color indicator --
COLOR=.TRUE. if an array of zip code colors is provided
COLOR=.FALSE. otherwise

MODIFY: LOGICAL
Color modification indicator --
MODIFY=.TRUE. if the array of zip code colors is allowed to be modified
MODIFY=.FALSE. otherwise

M: INTEGER*4
Number of zip codes

XCENT(M): REAL*4
Array containing x coordinates of centroids, sorted

```
            according to increasing x and y
YCENT(M):   REAL*4
            Array containing y coordinates of centroids, sorted
            according to increasing x and y
INDEX(M):   INTEGER*2
            Zip code index array --
            INDEX(I) is the original index of the zip code with
            centroid (XCENT(I),YCENT(I)) before sorting
PNT(M):     INTEGER*2
            Pointer array --
            PNT(I) points to the beginning of information on the
            zip code with centroid (XCENT(I),YCENT(I)) in the zip
            code boundary file
ZIPCLR(M): INTEGER*4
            Zip code color array --
            ABS(ZIPCLR(INDEX(I))) is the color (a number from 1 to
            9) of the zip code with centroid (XCENT(I),YCENT(I))
            (only input if COLOR=.TRUE.)
            Positive value: centroid drawn as asterisk
            Negative value: centroid drawn as small box

Output Variables --
IERR:       INTEGER*4
            Error flag --
            IERR=0 for normal return
            IERR=1 if an error was encountered
COLOR:      LOGICAL
            Set to .TRUE. if color array was created
ZIPCLR(M): INTEGER*4
            Modified zip code color array
            (same as input if MODIFY=.FALSE.)

Programs Called --
CHOICE
CLSGKS
GCLRWK
GRQCH
GSELNT
GSTXCI
GTX
MAPKEY
MENU11
MENU12
OPNGKS
STEMAP
ZIPMAP
ZOOMIN

Calling Programs --
PODMAP
SOLMAP
WRKMAP
```

PROGRAM DRIVER:

This program is the driver for the POD location system. It first initializes the report file STATEXX.REP and sets up data structures for storing zip code information. Then it uses GKS to write the top-level menu on the screen and allows the user to choose which phase of a problem to work on. The possible choices are:

(1) Exit.
(2) Display workload.
(3) Display or modify initial POD sites.
(4) Solve for optimal POD locations.
(5) Display optimal POD locations.

The driver terminates in one of two possible ways:

(1) The user chooses "F1 - EXIT" from the top menu. In this case, a file called EXITFILE.BAT is created which signals the batch program LOCATE.BAT to leave the POD location system.
(2) The user chooses "F4 - SOLVE FOR OPTIMAL POD LOCATIONS" from the top menu. In this case, LOCATE.BAT passes control to the solution algorithm SOLVER.COM. When the solution procedure is finished, control returns to DRIVER.EXE.

The following files must exist to run the POD location system.
(In the following, "XX" should be replaced by the state code number.)

(1) STATEXX.STE: STATE BOUNDARY FILE
(2) STATEXX.ZIP: ZIP CODE BOUNDARY FILE
(3) STATEXX.CEN: ZIP CODE CENTROID FILE
(4) STATEXX.POD: POD SITE FILE
(5) STATEXX.WRK: WORKLOAD FILE

(See the User's Manual for information on the format of these files.)

The following common block is required by the graphics package GKS:

```
COMMON /GRACOM/SIZE,INTARY.
```

SIZE is an INTEGER*4 variable set equal to 2500, and INTARY is an INTEGER*4 array of length 2500.

Source Code Location --
DRIVER.FOR

Definition of Variables --

CHDNR:	INTEGER*2 Choice device number (used to initialize function keys)
CHNR:	INTEGER*2 Choice number (number of function key selected)
COST(MAXPOD):	REAL*8 Work array containing POD costs for a zip code
CSQRFT(MAXPOD):	REAL*4 Array containing monthly rental costs per square foot for POD site office space
EX:	LOGICAL Flag indicating whether a file exists
FXCOST(MAXPOD):	REAL*4 Array containing opening/closing costs for POD sites

```
I:                INTEGER*4
                  Loop counter
IERR:             INTEGER*4
                  Error flag --
                  IERR=0 for normal return
                  IERR=1 if an error was encountered
INDEX(MAXM):      INTEGER*2
                  Zip code index array --
                  INDEX(I) is the original index of the zip code with
                  centroid (XCENT(I),YCENT(I)) in the zip code boundary
                  file
JCOEF(MAXPOD):    INTEGER*2
                  Work array
M:                INTEGER*4
                  Number of zip codes
MAXDIF:           INTEGER*4
                  Maximum possible number of non-unit travel difficulty
                  factors
MAXM:             INTEGER*4
                  Maximum possible number of zip codes
                  (initialized to be 1000)
MAXPOD:           INTEGER*4
                  Maximum possible number of POD sites
                  (initialized to be 85)
PAUSE:            INTEGER*4
                  Loop counter limit for error message pause
PNT(MAXM):        INTEGER*2
                  Pointer array --
                  PNT(I) points to the beginning of information on the
                  zip code with centroid (XCENT(I),YCENT(I)) in the zip
                  code boundary file
PODCLR(MAXM):     INTEGER*4
                  Zip code color array for POD site types
PODSET:           LOGICAL
                  Flag indicating whether POD types have been initialized
PODZIP(MAXPOD):   INTEGER*2
                  Array containing zip code indices of POD sites
STAT:             INTEGER*2
                  Status flag returned with function key --
                  STAT=0: function key not recognized
                  STAT=1: function key recognized successfully
STATE:            CHARACTER*2
                  Two-digit FIPS code number for state (must be entered
                  by user)
TRVDIF(MAXPOD):   REAL*4
                  Work array
WKID:             INTEGER*2
                  GKS workstation identifier
WORK(MAXPOD):     REAL*4
                  Work array
XCENT(MAXM):      REAL*4
                  Array containing x coordinates of centroids, sorted
                  according to increasing x and y
XPOD(MAXPOD):     REAL*4
```

Array containing x coordinates of centroids of POD sites

YCENT(MAXM): REAL*4
Array containing y coordinates of centroids, sorted according to increasing x and y

YPOD(MAXPOD): REAL*4
Array containing y coordinates of centroids of POD sites

ZIP5(MAXM): INTEGER*4
Array of five-digit zip codes, in original sequential order

ZIPCLR(MAXM): INTEGER*4
Zip code color array

ZIPIND(MAXM): INTEGER*2
Array of zip code indices, sorted by POD assignments

Programs Called --
CENSRT
CHOICE
CLSGKS
EXIT
GRQCH
OPNGKS
PODMAP
REPFIL
SOLMAP
SOLVE
TOPMNU
WRKMAP

SUBROUTINE ERSMNU:

This subroutine erases the current menu from the left side of the screen.

Source Code Location --
IRS.FOR

Programs Called --
BORDER
GFA
GQCNTN
GSELNT
STYLE

Calling Programs --
MAPKEY
ZOOMIN

SUBROUTINE EXIT(STATE):

This subroutine is executed when the user chooses "F1 - EXIT" from the top menu. It creates a batch file called EXITFILE.BAT which contains instructions for deleting work files that were created during the run. After this subroutine has been completed, control returns to the batch file LOCATE.BAT, which checks to see if EXITFILE.BAT exists, executes it, and leaves the system.

Source Code Location --
DRIVER.FOR

Input Variable --
STATE: CHARACTER*2
State code number

Programs Called --
None

Calling Program --
DRIVER

SUBROUTINE GACWK(WKID):

GKS routine -- "ACTIVATE WORKSTATION"
(See GKS manual for further information.)

Calling Program --
OPNGKS

SUBROUTINE GCLKS:

GKS routine -- "CLOSE KERNEL SYSTEM"
(See GKS manual for further information.)

Calling Program --
CLSGKS

SUBROUTINE GCLRWK(WKID,COFL):

GKS routine -- "CLEAR WORKSTATION"
(See GKS manual for further information.)

Calling Programs --
DISPLY
SPLTWN
ZOOMIN

SUBROUTINE GCLWK(WKID):

GKS routine -- "CLOSE WORKSTATION"
(See GKS manual for further information.)

Calling Program --
CLSGKS

SUBROUTINE GDAWK(WKID):

GKS routine -- "DEACTIVATE WORKSTATION"
(See GKS manual for further information.)

Calling Program --
CLSGKS

SUBROUTINE GEOFAC(DIFZIP,DIFPOD,DIFFAC,K1,NREC,STATE):

This subroutine allows the user to enter factors indicating the difficulty of travel between a zip code and a POD site. The values are stored in the file STATEXX.GDF.

Source Code Location --
DRIVER.FOR

Input Variables --
K1: INTEGER*4
Maximum number of non-unit travel difficulty factors
STATE: CHARACTER*2
State code number

Output Variables --
DIFZIP(NREC): INTEGER*4
Five-digit zip code array for use with travel difficulty factors
DIFPOD(NREC): INTEGER*4
POD zip code array for use with travel difficulty factors
DIFFAC(NREC): REAL*4
Array of travel difficulty factors --
DIFFAC(I) is the factor associated with zip code
DIFZIP(I) and POD site DIFPOD(I)
NREC: INTEGER*4
Number of non-unit travel difficulty factors

Programs Called --
None

Calling Program --
SOLVE

SUBROUTINE GETDAT(IYEAR,IMONTH,IDAY):

Professional FORTRAN routine -- "GET DATE"
(See Professional FORTRAN manual for further information.)

Calling Program --
REPFIL

SUBROUTINE GETTIM(IHOUR,IMIN,ISEC,IHUND):

Professional FORTRAN routine -- "GET TIME"
(See Professional FORTRAN manual for further information.)

Calling Program --
REPFIL

SUBROUTINE GFA(N,PX,PY):

GKS routine -- "FILL AREA"
(See GKS manual for further information.)

Calling Programs --
ERSMNU
MAPKEY
MENU20
MENU21
MENU22
MENU23
ZIPMAP

SUBROUTINE GINKS(NTYPES,WNAMES,WTYPES,VERNUM):

GKS routine -- "INITIALIZE GKS"
(See GKS manual for further information.)

Calling Program --
OPNGKS

SUBROUTINE GINLC(WKID,LCDNR,TNR,IPX,IPY,PET,XMIN,XMAX,YMIN,YMAX,LDR,DATREC):

GKS routine -- "INITIALIZE LOCATOR"
(See GKS manual for further information.)

Calling Programs --
CRSBOX
CURSOR

SUBROUTINE GOPKS(ERRFIL,SIZE):

GKS routine -- "OPEN KERNEL SYSTEM"
(See GKS manual for further information.)

Calling Program --
OPNGKS

SUBROUTINE GOPWK(WKID,CONID,WTYPE):

GKS routine -- "OPEN WORKSTATION"
(See GKS manual for further information.)

Calling Program --
OPNGKS

SUBROUTINE GPL(N,PX,PY):

GKS routine -- "POLYLINE"
(See GKS manual for further information.)

Calling Programs --
BOX
MAPKEY
MENU11
MENU12
MENU20
MENU21
MENU22
MENU23
STEMAP
TOPMNU
ZIPMAP

SUBROUTINE GPM(N,PX,PY):

GKS routine -- "POLYMARKER"
(See GKS manual for further information.)

Calling Programs --
MAPKEY
STEMAP
ZIPMAP

SUBROUTINE GPREC(IL,IA,RL,RA,SL,MSTR,STR,MLDR,ERRIND,LDR,DATREC):

GKS routine -- "PACK DATA RECORD"
(See GKS manual for further information.)

Calling Programs --
CRSBOX
CURSOR

SUBROUTINE GQCF(WTYPE,ERRIND,NCOLI,COLA,NPCI):

GKS routine -- "INQUIRE COLOR FACILITIES"
(See GKS manual for further information.)

Calling Programs --
SETCOL
STEMAP
STYLE
TOPMNU

SUBROUTINE GQCNTN(ERRIND,CTNR):

GKS routine -- "INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER"
(See GKS manual for further information.)

Calling Programs --
ERSMNU
MENU11
MENU12
MENU20
MENU21
MENU22
MENU23

SUBROUTINE GQMDS(WTYPE,ERRIND,DCUNIT,RX,RY,LX,LY):

GKS routine -- "INQUIRE MAXIMUM DISPLAY SURFACE SIZE"
(See GKS manual for further information.)

Calling Programs --
SPLTWN
WINDOW

SUBROUTINE GQNT(TNR,ERRIND,WINDOW,VIEWPT):

GKS routine -- "INQUIRE NORMALIZATION TRANSFORMATION"
(See GKS manual for further information.)

Calling Programs --
CRSBOX
MAPKEY

SUBROUTINE GQTXX(WKID,PX,PY,STR,ERRIND,CPX,CPY,TXEXPX,TXEXPY):

GKS routine -- "INQUIRE TEXT EXTENT"
(See GKS manual for further information.)

Calling Program --
SPLTWN

SUBROUTINE GRQCH(WKID,CHDNR,STAT,CHNR):

GKS routine -- "REQUEST CHOICE"
(See GKS manual for further information.)

Calling Programs --
DISPLY
DRIVER
MAPKEY

SUBROUTINE GRQLC(WKID,LCDNR,STAT,TNR,PX,PY):

GKS routine -- "REQUEST LOCATOR"
(See GKS manual for further information.)

Calling Programs --
CRSBOX
MAPKEY

SUBROUTINE GSCHM(WKID,CHDNR,MODE,ESW):

GKS routine -- "SET CHOICE MODE"
(See GKS manual for further information.)

Calling Program --
CHOICE

SUBROUTINE GSCR(WKID,CI,CR,CG,CG):

GKS routine -- "SET COLOR REPRESENTATION"
(See GKS manual for further information.)

Calling Program --
SETCOL

SUBROUTINE GSELNT(TNR):

GKS routine -- "SELECT NORMALIZATION TRANSFORMATION"
(See GKS manual for further information.)

Calling Programs --
DISPLY
ERSMNU
MAPKEY
MENU11
MENU12
MENU20
MENU21
MENU22
MENU23
SPLTWN
WINDOW
ZOOMIN

SUBROUTINE GSFACI(COLI):

GKS routine -- "SET FILL AREA COLOR INDEX"
(See GKS manual for further information.)

Calling Program --
STYLE

SUBROUTINE GSFAIS(INTS):

GKS routine -- "SET FILL AREA INTERIOR STYLE"
(See GKS manual for further information.)

Calling Program --
STYLE

SUBROUTINE GSFASI(STYLI):

GKS routine -- "SET FILL AREA STYLE INDEX"
(See GKS manual for further information.)

Calling Program --
STYLE

SUBROUTINE GSMK(MTYPE):

GKS routine -- "SET POLYMARKER TYPE"
(See GKS manual for further information.)

Calling Programs --
MAPKEY
STEMAP
ZIPMAP

SUBROUTINE GSPLCI(COLI):

GKS routine -- "SET POLYLINE COLOR INDEX"
(See GKS manual for further information.)

Calling Programs --
BORDER
MAPKEY
MENU11
MENU12
MENU20
MENU21
MENU22
MENU23
STEMAP
TOPMNU
ZIPMAP

SUBROUTINE GSPMCI(COLI):

GKS routine -- "SET POLYMARKER COLOR INDEX"
(See GKS manual for further information.)

Calling Programs --
MAPKEY
MATCH
STEMAP
ZIPMAP

SUBROUTINE GSTXCI(COLI):

GKS routine -- "SET TEXT COLOR INDEX"
(See GKS manual for further information.)

Calling Programs --
DISPLY
MAPKEY
MENU11

MENU12
MENU20
MENU21
MENU22
MENU23
TOPMNU
ZOOMIN

SUBROUTINE GSVP(TNR,XMIN,XMAX,YMIN,YMAX):

GKS routine -- "SET VIEWPORT"
(See GKS manual for further information.)

Calling Programs --
SPLTWN
WINDOW

SUBROUTINE GSWKVP(WKID,XMIN,XMAX,YMIN,YMAX):

GKS routine -- "SET WORKSTATION VIEWPORT"
(See GKS manual for further information.)

Calling Programs --
SPLTWN
WINDOW

SUBROUTINE GSWKWN(WKID,XMIN,XMAX,YMIN,YMAX):

GKS routine -- "SET WORKSTATION WINDOW"
(See GKS manual for further information.)

Calling Programs --
SPLTWN
WINDOW

SUBROUTINE GSWN(TNR,XMIN,XMAX,YMIN,YMAX):

GKS routine -- "SET WINDOW"
(See GKS manual for further information.)

Calling Programs --
SPLTWN
WINDOW

SUBROUTINE GTX(PX,PY,CHARS):

GKS routine -- "TEXT"
(See GKS manual for further information.)

Calling Programs --
DISPLY
MAPKEY
MENU11
MENU12
MENU20
MENU21
MENU22
MENU23
TOPMNU
ZOOMIN

**SUBROUTINE MAPKEY(IERR,MODIFY,TRN,PALETT,NCLRS,STATE,MENU,
M,XCENT,YCENT,INDEX,PNT,ZIPCLR):**

This subroutine displays a color key for the current zip code map. It allows the user to locate a zip code on the map with the cursor and have its five-digit name printed on the screen. (To do this, a search of the centroid array is done to find the closest centroid to the cursor position.) It may also allow the color of a zip code to be changed.

Source Code Location --
IRS.FOR

Input Variables --

MODIFY:	LOGICAL
	Color modification indicator --
	MODIFY=.TRUE. if the array of zip code colors is allowed to be modified
	MODIFY=.FALSE. otherwise
TRN:	INTEGER*2
	Transformation number
PALETT:	INTEGER*2
	Palette identifier (1 or 2)
NCLRS:	INTEGER*2
	Number of colors in menu
	(only used if MODIFY=.TRUE.)
STATE:	CHARACTER*2
	State code number
MENU:	INTEGER*2
	Menu type switch --
	MENU=0 for general coloring menu
	MENU=1 for menu of POD types
	MENU=2 for title of solution map
	MENU=3 for key to workload map
M:	INTEGER*4
	Number of zip codes

```
XCENT(M):   REAL*4
            Array containing x coordinates of centroids,
            sorted according to increasing x and y
YCENT(M):   REAL*4
            Array containing y coordinates of centroids,
            sorted according to increasing x and y
INDEX(M):   INTEGER*2
            Zip code index array, sorted corresponding to
            XCENT and YCENT arrays
PNT(M):     INTEGER*2
            Pointer array, sorted corresponding to XCENT and
            YCENT arrays
ZIPCLR(M):  INTEGER*4
            Zip code color array --
            Positive value: centroid drawn as asterisk
            Negative value: centroid drawn as small box

Output Variables --
IERR:       INTEGER*4
            Error flag --
            IERR=0 for normal return
            IERR=1 if an error was encountered
ZIPCLR(M):  INTEGER*4
            Zip code color array

Programs Called --
BORDER
CLSGKS
CURSOR
ERSMNU
GFA
GPL
GPM
GQNT
GRQCH
GRQLC
GSELNT
GSMK
GSPLCI
GSPMCI
GSTXCI
GTX
MENU20
MENU21
MENU22
MENU23
REVERS
SEARCH
STYLE

Calling Program --
DISPLY
```

SUBROUTINE MATCH(PALETT,INDEX):

This subroutine sets a polymarker color index which will match the corresponding fill area color index (for an enhanced graphics display only). (See the GKS manual for definitions of graphics terms.)

Source Code Location --
GKSUTIL.FOR

Input Variables --
PALETT: INTEGER*2
Palette identifier
INDEX: INTEGER*4
Index (between 1 and 16) of fill area color

Program Called --
GSPMCI

Calling Program --
STEMAP

SUBROUTINE MENU11:

This subroutine writes a menu of display options on the left side of the screen. It lists the meaning of the function keys after a state border map has been drawn.

Source Code Location --
IRS.FOR

Programs Called --
GPL
GQCNTN
GSELNT
GSPLCI
GSTXCI
GTX

Calling Program --
DISPLY

SUBROUTINE MENU12:

This subroutine writes a menu of display options on the left side of the screen. It lists the meaning of the function keys after a state zip code map has been drawn.

Source Code Location --
IRS.FOR

Programs Called --
GPL
GQCNTN
GSELNT
GSPLCI
GSTXCI
GTX

Calling Program --
DISPLY

SUBROUTINE MENU20:

This subroutine writes a color key on the left side of the screen. It lists the colors associated with the function keys.

Source Code Location --
IRS.FOR

Programs Called --
BORDER
BOX
GFA
GPL
GQCNTN
GSELNT
GSPLCI
GSTXCI
GTX
STYLE

Calling Program --
MAPKEY

SUBROUTINE MENU21:

This subroutine writes a color key on the left side of the screen. It lists the colors and function keys associated with different types of POD sites.

Source Code Location --
IRS.FOR

Programs Called --
BORDER
BOX
GFA
GPL
GQCNTN
GSELNT
GSPLCI

GSTXCI
GTX
STYLE

Calling Program --
MAPKEY

SUBROUTINE MENU22:

This subroutine writes a key for the solution map on the left side of the screen.

Source Code Location --
IRS.FOR

Programs Called --
BORDER
GFA
GPL
GQCNTN
GSELNT
GSPLCI
GSTXCI
GTX
STYLE

Calling Program --
MAPKEY

SUBROUTINE MENU23:

This subroutine writes a color key on the left side of the screen. It lists the colors and function keys used for displaying workload.

Source Code Location --
IRS.FOR

Programs Called --
BORDER
BOX
GFA
GPL
GQCNTN
GSELNT
GSPLCI
GSTXCI
GTX
STYLE

Calling Program --

MAPKEY

SUBROUTINE OPNGKS(WKID):

This subroutine opens GKS with one workstation of type "DISPLAY". Error messages will be written to a file called ERRORS.GKS. (See the GKS manual for definitions of graphics terms.)

Source Code Location --
GKSUTIL.FOR

Input Variable --
WKID: INTEGER*2
Workstation identifier

Programs Called --
GACWK
GINKS
GOPKS
GOPWK

Calling Programs --
DISPLY
DRIVER

SUBROUTINE OPTION(IO,DSTLIM,TXPWGT,IRSWGT,IMF,BMF):

This subroutine displays the default options for the cost calculation and allows the user to make changes.

Source Code Location --
DRIVER.FOR

Input Variables --
IO: INTEGER*4
Unit number defining where display will appear
DSTLIM: REAL*4
Default maximum travel distance allowed from zip code to POD
TXPWGT: REAL*4
Default weight (between 0 and 1) to be assigned to taxpayer travel costs
IRSWGT: REAL*4
Default weight (between 0 and 1) to be assigned to IRS travel costs
IMF(4): INTEGER*4
Default array indicating which IRS functions are to be included in the cost calculation for Individual Master File data --
IMF(1): Examination
IMF(2): Collection
IMF(3): Taxpayer Service

```
        IMF(4): Criminal Investigation
BMF(4): INTEGER*4
        Default array indicating which IRS functions are to be included
        in the cost calculation for Business Master File data --
        BMF(1): Examination
        BMF(2): Collection
        BMF(3): Taxpayer Service
        BMF(4): Criminal Investigation

Output Variables --
DSTLIM: REAL*4
        Maximum travel distance allowed from zip code to POD
TXPWGT: REAL*4
        Weight (between 0 and 1) to be assigned to taxpayer travel
        costs
IRSWGT: REAL*4
        Weight (between 0 and 1) to be assigned to IRS travel costs
IMF(4): INTEGER*4
        Array indicating which IRS functions are to be included in the
        cost calculation for IMF data
BMF(4): INTEGER*4
        Array indicating which IRS functions are to be included in the
        cost calculation for BMF data

Programs Called --
None

Calling Programs --
REPPRB
SOLVE
```

SUBROUTINE PODMAP(IERR,STATE,PAUSE,M,XCENT,YCENT,INDEX,PNT,ZIP5,ZIPCLR):

This subroutine initializes the display of a map of current and potential POD sites. It is executed when the user chooses "F3 - DISPLAY OR MODIFY INITIAL POD SITES" from the top menu. First it reads the file STATEXX.POD to find which zip codes are allowed to be POD sites and which zip codes currently are POD sites. Color #1 (background) is assigned to zip codes which cannot be POD sites, color #2 is assigned to potential POD sites, and color #3 is assigned to current POD sites. (Note: A negative color number indicates that the centroid will be drawn as a small box; otherwise the centroid is an asterisk.) Then these colors are passed in an array to the subroutine DISPLY, which draws the map and lets the user change the colors. The new colors are returned in the original array. Color #4 in the new array indicates a zip code which the user has made a fixed POD site.

```
Source Code Location --
DRIVER.FOR

Input Variables --
STATE:          CHARACTER*2
                State code number
```

```
PAUSE:          INTEGER*4
                Loop counter limit for error message pause
M:              INTEGER*4
                Number of zip codes
XCENT(M):       REAL*4
                Array containing x coordinates of centroids,
                sorted according to increasing x and y
YCENT(M):       REAL*4
                Array containing y coordinates of centroids,
                sorted according to increasing x and y
INDEX(M):       INTEGER*2
                Zip code index array --
                INDEX(I) is the original index of the zip code with
                centroid (XCENT(I),YCENT(I)) before sorting
PNT(M):         INTEGER*2
                Pointer array --
                PNT(I) points to the beginning of information on the
                zip code with centroid (XCENT(I),YCENT(I)) in the
                zip code boundary file
ZIP5(M):        INTEGER*4
                Array of five-digit zip codes, in original sequential
                order

Output Variables --
IERR:           INTEGER*4
                Error flag --
                IERR=0 for normal return
                IERR=1 if an error was encountered
ZIPCLR(M):      INTEGER*4
                Zip code color array for POD site types

Program Called --
DISPLY

Calling Program --
DRIVER
```

SUBROUTINE REPFIL(STATE):

This subroutine initializes the report file. Either a new report file is created or information is appended to an existing report file. A report header is written giving the date and time. The report file is assigned to unit 12, which remains open after returning from the subroutine.

```
Source Code Location --
DRIVER.FOR

Input Variable --
STATE: CHARACTER*2
       State code number

Programs called --
```

GETDAT
GETTIM

Calling Program --
DRIVER

SUBROUTINE REPPRB(M,ZIPCLR,ZIP5,DSTLIM,IRSWT,TXPWT,IMFFNC,BMFFNC,NREC, DIFZIP,DIFPOD,DIFFAC,FXCOST,CSQRFT,SQFT,MILCST,IRSFCT, TXPFCT):

This subroutine writes information on the problem initialization into the report file.

Source Code Location --
DRIVER.FOR

Input Variables --

M:	INTEGER*4 Number of zip codes
ZIPCLR(M):	INTEGER*4 Zip code color array for POD site types
ZIP5(M):	INTEGER*4 Array of five-digit zip codes
DSTLIM:	REAL*4 Maximum travel distance allowed from zip code to POD
IRSWT:	REAL*4 Weight assigned to IRS travel costs
TXPWT:	REAL*4 Weight assigned to taxpayer travel costs
IMFFNC(4):	INTEGER*4 Array indicating which IRS functions are included in the cost calculation for Individual Master File data
BMFFNC(4):	INTEGER*4 Array indicating which IRS functions are included in the cost calculation for Business Master File data
NREC:	INTEGER*4 Number of non-unit travel difficulty factors
DIFZIP(NREC):	INTEGER*4 Five-digit zip code array for use with travel difficulty factors
DIFPOD(NREC):	INTEGER*4 POD zip code array for use with travel difficulty factors
DIFFAC(NREC):	REAL*4 Array of travel difficulty factors
FXCOST(NPOD):	REAL*4 Array containing fixed costs of POD sites
CSQRFT(NPOD):	REAL*4 Array containing monthly rental costs per square foot for POD site office space
SQFT:	REAL*4 Office space required per IMF return (square feet)
MILCST:	REAL*4

```
                Mileage cost ($/mile)
IRSFCT(16):     REAL*4
                Array of IRS trip factors
TXPFCT(16):     REAL*4
                Array of taxpayer trip factors

Programs Called --
OPTION

Calling Program --
SOLVE
```

SUBROUTINE REPSOL(COST,NZIP,ZIPIND,NPOD,ENDPNT,ZIP5):

This subroutine writes information on the problem solution into the report file.

```
Source Code Location --
DRIVER.FOR

Input Variables --
COST:           REAL*8
                Total cost of assigning zip codes to POD sites
NZIP:           INTEGER*4
                Number of zip codes in solution file
ZIPIND(NZIP):   INTEGER*2
                Array containing zip code indices from the solution file,
                sorted according to POD assignments
NPOD:           INTEGER*4
                Number of POD sites in solution
ENDPNT(NPOD):   INTEGER*2
                Array of POD pointers --
                ZIPIND(ENDPNT(I)) is the last zip code index in ZIPIND
                assigned to the Ith POD site
ZIP5(M):        INTEGER*4
                Array of five-digit zip codes

Programs Called --
None

Calling Program --
SOLMAP
```

FUNCTION REVERS(X,MIN,MAX):

This function reverses the direction of a coordinate axis. (This is necessary because the x coordinates in the Ganesa files increase from right to left, but GKS assumes that they increase from left to right.)

```
Source Code Location --
```

```
IRS.FOR

Input Variables --
X:      REAL*4
        Value of coordinate to be reversed
MIN:    REAL*4
        Minimum value of coordinate axis
MAX:    REAL*4
        Maximum value of coordinate axis

Output Variable --
REVERS: REAL*4
        Reversed value of X

Programs Called --
None

Calling Programs --
MAPKEY
STEMAP
ZIPMAP
ZOOMIN
```

SUBROUTINE SEARCH(XPT,YPT,M,X,Y,POS):

This subroutine searches the list of sorted centroids to find the centroid closest to a given point. A binary search is done to find the nearest x coordinate, and then a sequential search is done to find the nearest y coordinate.

```
Source Code Location --
IRS.FOR

Input Variables --
XPT:  REAL*4
      X coordinate of point to be located
YPT:  REAL*4
      Y coordinate of point to be located
M:    INTEGER*4
      Number of zip codes
X(M): REAL*4
      Array containing x coordinates of centroids,
      sorted according to increasing x and y
Y(M): REAL*4
      Array containing y coordinates of centroids,
      sorted according to increasing x and y

Output Variable --
POS:  INTEGER*4
      Position of (XPT,YPT) in centroid list
      (i.e., (XPT,YPT)=(X(POS),Y(POS)))
      POS=0 if point was not found
```

Programs Called --
None

Calling Program --
MAPKEY

SUBROUTINE SETCOL(WKID,PALETT):

This subroutine defines the colors to be used by the display device. The type of display device in use is determined by calling the GKS routine GQCF which returns the number of colors available on the current device driver. For a medium-resolution four-color display, this subroutine sets palette two with a blue background. For an enhanced color display, it sets sixteen colors in one of two possible palettes. (See the GKS manual for definitions of graphics terms.)

Source Code Location --
GKSUTIL.FOR

Input Variables --
WKID: INTEGER*2
Workstation identifier
PALETT: INTEGER*2
Palette identifier (1 or 2) --
PALETT=1: 16 distinct colors
PALETT=2: colors 4 through 9 are shades of green
(PALETT is not used in medium resolution mode)

Programs Called --
GQCF
GSCR

Calling Programs --
SPLTWN
WINDOW

SUBROUTINE SOLMAP(IERR,STATE,PAUSE,M,XCENT,YCENT,INDEX,PNT,ZIPCLR,ZIP5, ZIPIND,ENDPNT):

This subroutine initializes the display of a solution map. It is executed when the user chooses "F5 - DISPLAY OPTIMAL POD LOCATIONS" from the top menu. It checks the solution file STATEXX.SOL to see whether the graph-coloring algorithm was used by the solver. (This algorithm is used only if an input file is present giving zip code adjacency information.) If graph-coloring was not used, then only zip codes which are POD sites (as determined by the solution) will be colored on the map (using color #3); all other zip codes will be in the background color (color #1). (Note: A negative color number indicates that the centroid will be drawn as a small box; otherwise the centroid is an asterisk.) If graph-coloring was used, then a color for each zip code is read from the solution file (the color

number is increased by one to avoid using the background color). The resulting map will show a POD and its assigned zip codes all in the same color. The subroutine REPSOL is called to write solution information in the report file and the subroutine DISPLY is called to draw the map.

```
Source Code Location --
DRIVER.FOR

Input Variables --
STATE:          CHARACTER*2
                State code number
PAUSE:          INTEGER*4
                Loop counter limit for error message pause
M:              INTEGER*4
                Number of zip codes
XCENT(M):       REAL*4
                Array containing x coordinates of centroids,
                sorted according to increasing x and y
YCENT(M):       REAL*4
                Array containing y coordinates of centroids,
                sorted according to increasing x and y
INDEX(M):       INTEGER*2
                Zip code index array --
                INDEX(I) is the original index of the zip code with
                centroid (XCENT(I),YCENT(I)) before sorting
PNT(M):         INTEGER*2
                Pointer array --
                PNT(I) points to the beginning of information on the
                zip code with centroid (XCENT(I),YCENT(I)) in the
                zip code boundary file
ZIP5(M):        INTEGER*4
                Array of five-digit zip codes

Output Variables --
IERR:           INTEGER*4
                Error flag --
                IERR=0 for normal return
                IERR=1 if an error was encountered
ZIPCLR(M):      INTEGER*4
                Zip code color array
ZIPIND(NZIP):   INTEGER*2
                Array of zip code indices, sorted by POD assignments
ENDPNT(NPOD):   INTEGER*2
                Array of POD pointers --
                ZIPIND(ENDPNT(I)) is the last zip code index in ZIPIND
                assigned to the Ith POD site

Program Called --
DISPLY
REPSOL

Calling Program --
DRIVER
```

SUBROUTINE SOLVE(IERR,STATE,PAUSE,MAXPOD,PODZIP,XPOD,YPOD,M,HOLD,FXCOST, CSQRFT,TRVDIF,ZIPCLR,ZIP5,JCOEF,COST,MAXDIF,DIFZIP,DIFPOD, DIFFAC):

This subroutine allows the user to set parameters to initialize a location problem and then calculates costs which are passed to the solution procedure SOLVER.COM. It is executed when the user chooses "F4 - SOLVE FOR OPTIMAL POD LOCATIONS" from the top menu. First, opening/closing costs and office rental costs are read from the file STATEXX.POD. If any POD sites exist which are not in the file, the user is asked to enter the costs. Then several parameters such as cost per mile, maximum travel distance, and trip factors are read from the file STATEXX.WRK. The user is allowed to set weights for IRS and taxpayer costs, to turn categories of workload on or off, to change the distance limit, and to set travel difficulty factors. The parameter settings are summarized in the report file STATEXX.REP.

After all parameters have been set, the program proceeds to calculate costs of assigning zip codes to POD sites. These costs are passed to the PASCAL solution procedure in the unformatted sequential file STATEXX.DBL. For each zip code, the program finds all possible POD sites (current, potential and fixed) within the specified distance limit and calculates the cost of each assignment using the function COSTFN. The assignments are then sorted in order of decreasing cost; if a zip code is a possible POD site, its cost is last in the list. The list of costs and POD sites for a particular zip code is written as one record of the file STATEXX.DBL. The type of the zip code (i.e., whether it is a potential POD site, a fixed POD site, etc.) is also included in the record. The record is padded with zeroes to make the file readable by a PASCAL program. If, for some zip code, there are no possible POD sites within the distance limit, an error message will appear on the screen. Otherwise, the subroutine is exited and control passes to the solution procedure. (The solution algorithm is described in "The Internal Revenue Service Post-of-Duty Location Modeling System: Programmer's Manual for PASCAL Solver".)

Source Code Location --
DRIVER.FOR

Input Variables --

STATE:	CHARACTER*2 State code number
PAUSE:	INTEGER*4 Loop counter limit for error message pause
MAXPOD:	INTEGER*4 Maximum possible number of POD sites
M:	INTEGER*4 Number of zip codes
ZIPCLR(M):	INTEGER*4 Zip code color array for POD site types
ZIP5(M):	INTEGER*4 Array of five-digit zip codes, in original sequential order
MAXDIF:	INTEGER*4 Maximum number of non-unit travel difficulty factors

```
Output Variables --
IERR:          INTEGER*4
               Error flag --
               IERR=0 for normal return
               IERR=1 if an error was encountered
PODZIP(NPOD): INTEGER*4
               Array containing zip code indices of POD sites
XPOD(NPOD):    REAL*4
               Array containing x coordinates of centroids of
               POD sites
YPOD(NPOD):    REAL*4
               Array containing y coordinates of centroids of
               POD sites
HOLD(NPOD):    REAL*4
               Work array
FXCOST(NPOD): REAL*4
               Array containing fixed costs (opening/closing costs) of
               POD sites
CSQRFT(NPOD): REAL*4
               Array containing monthly rental costs per square foot for
               POD site office space
TRVDIF(NPOD): REAL*4
               Work array containing travel difficulty factors from a
               zip code to all possible POD sites
JCOEF(NPOD):   INTEGER*2
               Work array containing POD indices for a zip code
COST(NPOD):    REAL*8
               Work array containing POD costs for a zip code
DIFZIP(NREC): INTEGER*4
               Five-digit zip code array for use with travel difficulty
               factors
DIFPOD(NREC): INTEGER*4
               POD zip code array for use with travel difficulty factors
DIFFAC(NREC): REAL*4
               Array of travel difficulty factors --
               DIFFAC(I) is the factor associated with zip code
               DIFZIP(I) and POD site DIFPOD(I)

Programs Called --
COSTFN
GEOFAC
OPTION
REPPRB
SORT

Calling Program --
DRIVER
```

SUBROUTINE SORT(M,X,Y,INDEX,PNT):

This subroutine uses a heapsort algorithm to sort the centroids according to increasing x and y coordinates. (It is also used to sort other arrays.)

For a description of the heapsort algorithm, see, for example, An Introduction to Data Structures with Applications by J. P. Tremblay and P. G. Sorenson (McGraw-Hill, 1976, p. 475).

Source Code Location --
IRS.FOR

Input Variables --
M: INTEGER*4
Number of zip codes
X(M): REAL*4
Array containing x coordinates of centroids
Y(M): REAL*4
Array containing y coordinates of centroids
INDEX(M): INTEGER*2
Zip code index array --
INDEX(I)=I on input
PNT(M): INTEGER*2
Pointer array --
PNT(I) points to the beginning of information on the Ith zip code in the zip code boundary file

Output Variables --
X(M): REAL*4
Array containing x coordinates of centroids, sorted according to increasing x and y
Y(M): REAL*4
Array containing y coordinates of centroids, sorted according to increasing x and y
INDEX(M): INTEGER*2
Zip code index array --
INDEX(I) is the original index of the zip code with centroid (XCENT(I),YCENT(I)) before sorting
PNT(M): INTEGER*2
Pointer array, sorted corresponding to XCENT and YCENT arrays

Programs Called --
None

Calling Programs --
CENSRT
SOLVE

SUBROUTINE SPLTWN(WKID,RTRN,PALETT,XMIN,XMAX,YMIN,YMAX):

This subroutine sets up a split window and viewport (designed for a menu on the left side and a map on the right side). World coordinates for the left side must be between 0 and 1 and use transformation number 7. The width of the left window is set to be the width of eight characters. World coordinates and a transformation number for the right side must be specified

by the user, and the aspect ratio of the data will be preserved. (See the GKS manual for definitions of graphcis terms.)

Source Code Location --
GKSUTIL.FOR

Input Variables --
WKID: INTEGER*2
Workstation identifier
RTRN: INTEGER*2
Right transformation number (between 1 and 6)
PALETT: INTEGER*2
Palett identifier (1 or 2)
XMIN: REAL*4
Smallest x value of right world coordinates
XMAX: REAL*4
Largest x value of right world coordinates
YMIN: REAL*4
Smallest y value of right world coordinates
YMAX: REAL*4
Largest y value of right world coordinates

Programs Called --
BORDER
GCLRWK
GQMDS
GQTXX
GSELNT
GSVP
GSWKVP
GSWKWN
GSWN
SETCOL
WINDOW

Calling Programs --
STEMAP
ZIPMAP

SUBROUTINE STEMAP(IERR,TRN,PALETT,STATE,XMIN,XMAX,YMIN,YMAX, M,XCENT,YCENT,INDEX,ZIPCLR):

This subroutine draws a state map, showing the outline of the state and centroids of zip codes. (Zip code boundaries are not drawn.) The state border is read from the file STATEXX.STE.

Source Code Location --
IRS.FOR

Input Variables --
TRN: INTEGER*2
Transformation number

```
PALETT:     INTEGER*2
            Palette identifier (1 or 2)
STATE:      CHARACTER*2
            State code number
M:          INTEGER*4
            Number of zip codes
XCENT(M):   REAL*4
            Array containing x coordinates of centroids
YCENT(M):   REAL*4
            Array containing y coordinates of centroids
INDEX(M):   INTEGER*2
            Zip code index array
ZIPCLR(M):  INTEGER*4
            Zip code color array --
            Positive value: centroid drawn as asterisk
            Negative value: centroid drawn as small box

Output Variables --
IERR:       INTEGER*4
            Error flag --
            IERR=0 for normal return
            IERR=1 if an error was encountered
XMIN:       REAL*4
            X coordinate of lower left corner of map
XMAX:       REAL*4
            X coordinate of upper right corner of map
YMIN:       REAL*4
            Y coordinate of lower left corner of map
YMAX:       REAL*4
            Y coordinate of upper right corner of map

Programs Called --
BEEP
CLSGKS
GPL
GPM
GQCF
GSMK
GSPLCI
GSPMCI
MATCH
REVERS
SPLTWN

Calling Program --
DISPLY
```

SUBROUTINE STYLE(IERR,WKID,PALETT,INDEX):

This subroutine sets one of sixteen area-filling interior styles. For a medium-resolution four-color display, the style is either a solid color or a

cross-hatch pattern. For an enhanced color display, the style is always a solid color. (See the GKS manual for definitions of graphics terms.)

```
Source Code Location --
GKSUTIL.FOR

Input Variables --
WKID:   INTEGER*2
        Workstation identifier
PALETT: INTEGER*2
        Palette identifier (1 or 2) --
        PALETT=1: 16 distinct colors
        PALETT=2: 6 shades of green plus 10 other colors
        (PALETT is not used in medium resolution mode)
INDEX:  INTEGER*4
        Index (between 1 and 16) of desired interior style

Output Variable --
IERR:   INTEGER*4
        Error flag --
        IERR=0 for normal return
        IERR=1 if an error was encountered

Programs Called --
CLSGKS
GQCF
GSFACI
GSFAIS
GSFASI

Calling Programs --
ERASMNU
MAPKEY
MENU20
MENU21
MENU22
MENU23
ZIPMAP
```

SUBROUTINE TOPMNU:

This subroutine writes the text for the top-level menu on the screen. The format of the menu is slightly different depending on whether a regular color display or an enhanced color display is being used. This is determined by calling the GKS routine GQCF which finds the number of colors available on the current device driver.

```
Source Code Location --
DRIVER.FOR

Programs Called --
BORDER
```

```
GPL
GQCF
GSPLCI
GSTXCI
GTX
WINDOW

Calling Program --
DRIVER
```

SUBROUTINE WINDOW(WKID,TRN,PALETT,XMIN,XMAX,YMIN,YMAX):

This subroutine sets a window and a viewport in a way that preserves the aspect ratio of the world coordinates. (See the GKS manual for definitions of graphics terms.)

```
Source Code Location --
GKSUTIL.FOR

Input Variables --
WKID:   INTEGER*2
        Workstation identifier
TRN:    INTEGER*2
        Transformation number (between 1 and 7)
PALETT: INTEGER*2
        Palette identifier (1 or 2)
XMIN:   REAL*4
        Smallest x value of world coordinates
XMAX:   REAL*4
        Largest x value of world coordinates
YMIN:   REAL*4
        Smallest y value of world coordinates
YMAX:   REAL*4
        Largest y value of world coordinates

Programs Called --
GQMDS
GSELNT
GSVP
GSWKVP
GSWKWN
GSWN
SETCOL

Calling Programs --
SPLTWN
TOPMNU
```

SUBROUTINE WRKMAP(IERR,STATE,PAUSE,M,XCENT,YCENT,INDEX,PNT,ZIPCLR):

This subroutine initializes the display of a workload map. It is executed when the user chooses "F2 - DISPLAY WORKLOAD" from the top menu. It first asks the user to choose the category of workload to be displayed and then sets a series of sixteen switches corresponding to the desired columns of the workload file STATEXX.WRK. These columns are read and summed for each zip code. Then the workloads are divided into six equally spaced ranges. Each zip code is assigned a color according to the range its workload falls in. The colors are passed in an array to the subroutine DISPLY which draws the map.

```
Source Code Location --
DRIVER.FOR

Input Variables --
STATE:      CHARACTER*2
            State code number
PAUSE:      INTEGER*4
            Loop counter limit for error message pause
M:          INTEGER*4
            Number of zip codes
XCENT(M):   REAL*4
            Array containing x coordinates of centroids,
            sorted according to increasing x and y
YCENT(M):   REAL*4
            Array containing y coordinates of centroids,
            sorted according to increasing x and y
INDEX(M):   INTEGER*2
            Zip code index array --
            INDEX(I) is the original index of the zip code with
            centroid (XCENT(I),YCENT(I)) before sorting
PNT(M):     INTEGER*2
            Pointer array --
            PNT(I) points to the beginning of information on the
            zip code with centroid (XCENT(I),YCENT(I)) in the zip
            code boundary file

Output Variables --
IERR:       INTEGER*4
            Error flag --
            IERR=0 for normal return
            IERR=1 if an error was encountered
ZIPCLR(M):  INTEGER*4
            Zip code color array for workload

Program Called --
DISPLY

Calling Program --
DRIVER
```

```
SUBROUTINE ZIPMAP(IERR,TRN,PALETT,STATE,ZOOM,COLOR,XMIN,XMAX,
                  YMIN,YMAX,M,XCENT,YCENT,INDEX,PNT,ZIPCLR):
```

This subroutine draws a zip code map either for a full state or for a smaller area. Centroids are included, and the zip codes are colored if specified. The zip code boundaries are read from the file STATEXX.ZIP. If the whole state is not to be drawn, but only a smaller region specified by a zoom rectangle, a search is done first to find the first and last zip codes (as given in the sorted centroid list) included in the rectangle.

Source Code Location --
IRS.FOR

Input Variables --

TRN: INTEGER*2
Transformation number

PALETT: INTEGER*2
Palette identifier (1 or 2)

STATE: CHARACTER*2
State code number

ZOOM: LOGICAL
Zoom indicator --
ZOOM=.FALSE. if entire state is to be drawn
ZOOM=.TRUE. if only part of state is to be drawn

COLOR: LOGICAL
Color indicator --
COLOR=.TRUE. if zip codes are to be colored as they are drawn
COLOR=.FALSE. if just zip code boundaries are to be drawn

XMIN: REAL*4
X coordinate of lower left corner of zoom rectangle (used only if ZOOM=.TRUE.)

XMAX: REAL*4
X coordinate of upper right corner of zoom rectangle (used only if ZOOM=.TRUE.)

YMIN: REAL*4
Y coordinate of lower left corner of zoom rectangle (used only if ZOOM=.TRUE.)

YMAX: REAL*4
Y coordinate of upper right corner of zoom rectangle (used only if ZOOM=.TRUE.)

M: INTEGER*4
Number of zip codes

XCENT(M): REAL*4
Array containing x coordinates of centroids

YCENT(M): REAL*4
Array containing y coordinates of centroids

INDEX(M): INTEGER*2
Zip code index array

PNT(M): INTEGER*2
Pointer array --
PNT(I) points to the beginning of information on the zip code with centroid (XCENT(I),YCENT(I)) in the zip code boundary file

ZIPCLR(M): INTEGER*4
Zip code color array --

```
                (colors used only if COLOR=.TRUE.)
                Positive value: centroid drawn as asterisk
                Negative value: centroid drawn as small box

     Output Variables --
     IERR:      INTEGER*4
                Error flag --
                IERR=0 for normal return
                IERR=1 if an error was encountered
     XMIN:      REAL*4
                X coordinate of lower left corner of map
                (same as input XMIN if ZOOM=.TRUE.)
     XMAX:      REAL*4
                X coordinate of upper right corner of map
                (same as input XMAX if ZOOM=.TRUE.)
     YMIN:      REAL*4
                Y coordinate of lower left corner of map
                (same as input YMIN if ZOOM=.TRUE.)
     YMAX:      REAL*4
                Y coordinate of upper right corner of map
                (same as input YMAX if ZOOM=.TRUE.)

     Programs Called --
     BEEP
     BORDER
     CLSGKS
     GFA
     GPL
     GPM
     GSMK
     GSPLCI
     GSPMCI
     REVERS
     SPLTWN
     STYLE

     Calling Programs --
     DISPLY
     ZOOMIN
```

SUBROUTINE ZOOMIN(IERR,TRN,PALETT,STATE,XMIN,XMAX,YMIN,YMAX,
M,XCENT,YCENT,INDEX,PNT,ZIPCLR):

This subroutine allows the user to draw a box with the cursor on a previously drawn map and then zooms in on the boxed area.

```
     Source Code Location --
     IRS.FOR

     Input Variables --
     TRN:       INTEGER*2
                Transformation number
```

```
PALETT:     INTEGER*2
            Palette identifier (1 or 2)
STATE:      CHARACTER*2
            State code number
XMIN:       REAL*4
            X coordinate of lower left corner of current map
XMAX:       REAL*4
            X coordinate of upper right corner of current map
YMIN:       REAL*4
            Y coordinate of lower left corner of current map
YMAX:       REAL*4
            Y coordinate of upper right corner of current map
M:          INTEGER*4
            Number of zip codes
XCENT(M):   REAL*4
            Array containing x coordinates of centroids
YCENT(M):   REAL*4
            Array containing y coordinates of centroids
INDEX(M):   INTEGER*2
            Zip code index array
PNT(M):     INTEGER*2
            Pointer array --
            PNT(I) points to the beginning of information on the
            zip code with centroid (XCENT(I),YCENT(I)) in the zip
            code boundary file
ZIPCLR(M): INTEGER*4
            Zip code color array

Output Variables
IERR:       INTEGER*4
            Error flag --
            IERR=0 for no return
            IERR=1 if an error was encountered
TRN:        INTEGER*2
            Transformation number
XMIN:       REAL*4
            X coordinate of lower left corner of zoom region
XMAX:       REAL*4
            X coordinate of upper right corner of zoom region
YMIN:       REAL*4
            Y coordinate of lower left corner of zoom region
YMAX:       REAL*4
            Y coordinate of upper right corner of zoom region

Programs Called --
CRSBOX
ERSMNU
GCLRWK
GSELNT
GSTXCI
GTX
REVERS
ZIPMAP
```

Calling Program --
DISPLY

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 86-3473	2. Performing Organ. Report No.	3. Publication Date FEBRUARY 1987

4. TITLE AND SUBTITLE

The Internal Revenue Service Post-of-Duty Location Modeling System - Programmer's Manual for Fortran Driver

5. AUTHOR(S)

Paul D. Domich, Richard H. F. Jackson, Marjorie A. McClain

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

The Research Division
U. S. Internal Revenue Service
1201 E Street
Washington, DC 20224

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

This report is a programmer's manual for a microcomputer package which was designed by the National Bureau of Standards to assist the Internal Revenue Service in choosing locations for its posts-of-duty which will minimize costs to the IRS and to the taxpayer. The package was written in two sections of code, one in FORTRAN and the other in PASCAL. This manual describes the FORTRAN driver which handles graphics displays and controls input and output for the solution procedures.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

facility location, interactive graphics, personal computer, microcomputer, Graphical Kernel System (GKS)

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

55

15. Price

$13.95